Paper 021-2007

ODS Options and SAS® Stored Processes

Cynthia L. Zender, SAS Institute Inc., Cary, NC

ABSTRACT

Are you using SAS Stored Processes in a SAS Intelligence Platform client-server environment? Do you want to convert existing SAS programs for use as SAS Stored Processes? This paper provides concrete examples of how to convert existing SAS programs to stored processes that can be executed from selected SAS Intelligence Platform client applications. Topics in this paper include: basic report program conversion, conversion of SAS/GRAPH® programs, use of macro variables in program conversion, streaming versus transient output from stored processes, and permanent result packages. Special emphasis is given to showing code both before and after conversion—especially the use of Output Delivery System (ODS) options. SAS/GRAPH options are also illustrated with before-and-after examples. In addition, there is some discussion of how different client applications accept the options and render output.

INTRODUCTION

The purpose of this paper is to show how an existing SAS program that uses `ODS RTF`, `ODS PDF`, or `ODS HTML` can be converted to execute as a SAS stored process. For the sake of brevity and ease of explanation, the ODS options and user-supplied parameters have been kept as simple as possible. Data from the `SASHELP` library has been used for the demonstration programs. However, the full text of the two "starter" programs is included in the Appendix. If you are interested in the converted programs or stored processes that were derived from these starter programs, you can download them from the Web site listed in the "Resources" section at the end of the paper.

Security and authorization discussions fall outside the scope of this paper. Not every stored process author will have the ability to alter authorization permissions or to move stored processes around in a DEV/TEST/PROD environment. The paper does not discuss using the SAS® Enterprise Guide® Stored Process Wizard. The main focus of this paper is the code changes that must be implemented to use ODS options and SAS/GRAPH options with SAS Stored Processes.

BASIC REPORT PROGRAM CONVERSION STEPS

The basic steps for program conversion are

1. Have a working SAS program
2. Convert the SAS program to be a stored process by adding the `*ProcessBody;` comment and `%STPBEGIN;`/`%STPEND;` macro invocations in place of the ODS invocation statements
3. Register the stored process metadata relevant to the stored process program
4. Test your stored process in all client applications from which it will be executed.

Part of the conversion process in Step 2 and the registration process in Step 3 involves understanding how user-supplied parameter values are treated by stored processes and how ODS options need to be supplied in order to affect stored process results. Two programs were converted to stored processes for this paper. The first program contains a PROC REPORT step and a PROC TABULATE step. We will use the **DEMO1_TABLE.SAS** program to talk about simple conversion methods and we'll talk about the second program (**DEMO2_HTML.SAS**) when we discuss some of the more complex issues related to stored process conversion.

STEP 1: HAVE A WORKING SAS PROGRAM

The first program, **DEMO1_TABLE.SAS**, contains a PROC REPORT step and a PROC TABULATE step and is outlined below.

```
 %let wantreg=Canada;

options nodate nonumber missing='0' orientation=landscape;

ods rtf file='c:\temp\demo1.rtf' bodytitle startpage=no keepn notoc_data;
ods pdf file='c:\temp\demo1.pdf' bookmarkgen=no compress=9 startpage=no;
ods html file='c:\temp\demo1.html' style=sasweb rs=none;
ods escapechar='^';
  proc sort data=sashelp.shoes out=shoes;  by Region;
        where Region = "&wantreg";
  run;

**  proc report code;
**  proc tabulate code;

ods _all_ close;
```

The macro variable, &WANTREG, is used at the highlighted places in the code to limit the program execution to only one REGION from the SASHELP.SHOES data set. In addition, the code uses a common ODS technique that produces three output files, HTML, RTF, and PDF, with only one execution of the procedure code. Each ODS statement uses options that are specific to that destination.

The ODS options (other than the FILE= option) used in the program are shown in the table below. For an explanation of the options, refer to the Appendix.

RTF	PDF	HTML
BODYTITLE	BOOKMARKGEN=NO	STYLE=SASWEB
STARTPAGE=NO	STARTPAGE=NO	RS=NONE
KEEPN	COMPRESS=9	
NOTOC_DATA	TEXT=	

The output from the **DEMO1_TABLE.SAS** program is shown below (Figures 1, 2, and 3). You can see that the SASWEB style was used for the HTML file; the BODYTITLE option was used for the RTF file; and ODS TEXT= was used for the PDF file. The other options, like COMPRESS= or RS=NONE are not immediately obvious, but could be detected if you compared the uncompressed PDF file size with the file size of this output, or if you looked at the HTML source file to see that the appropriate record separators were put at the end of each line of HTML source code. You can also see that the LANDSCAPE orientation, which was specified in an OPTIONS statement, was used for the RTF and PDF files.

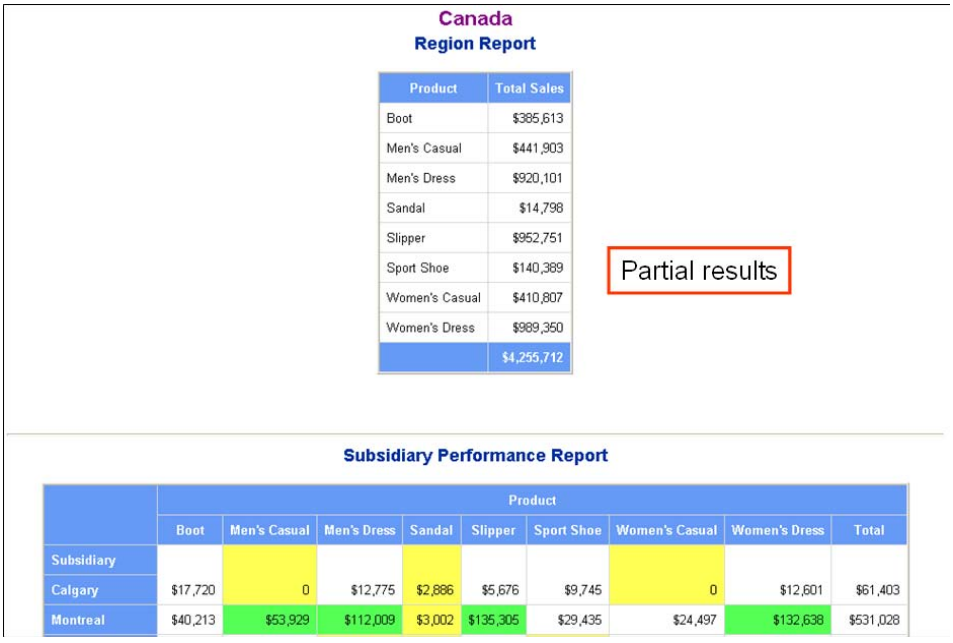

Figure 1: Partial HTML Output Viewed in Internet Explorer

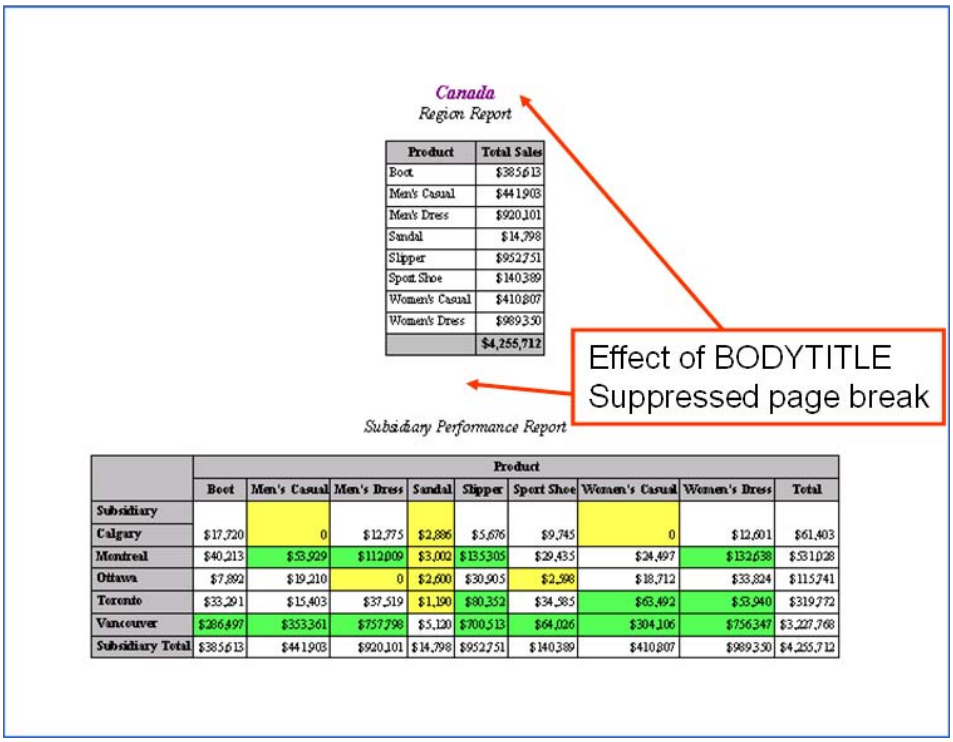

Figure 2: RTF Output in Print Preview Mode of Microsoft Word

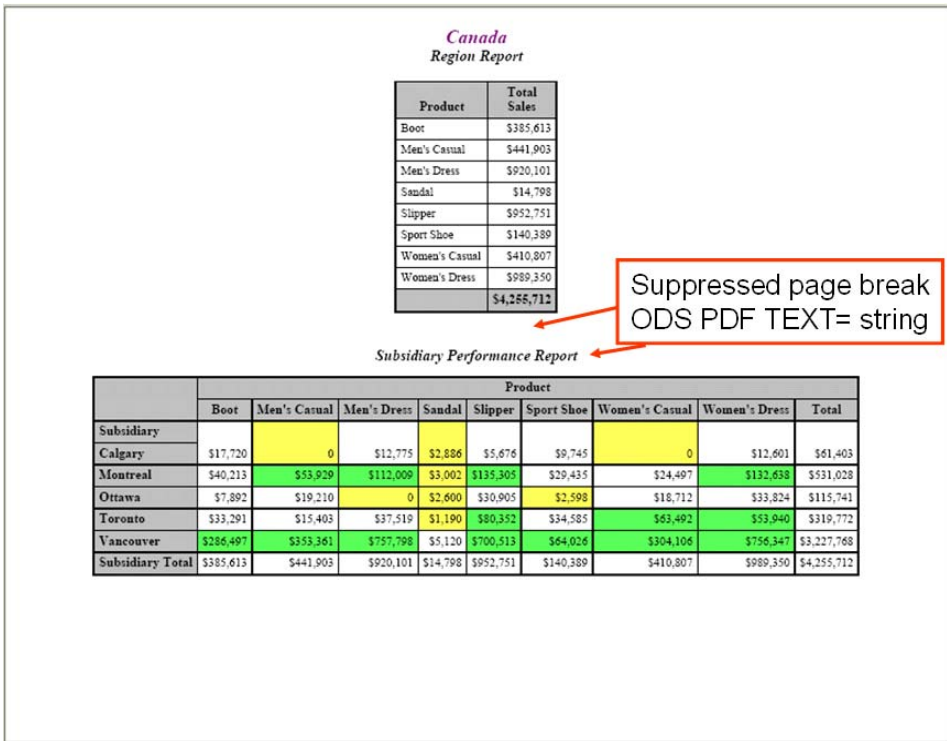

Figure 3: PDF Output Viewed in Adobe Acrobat Reader 7

Now that we have a working SAS program, Step 1 is satisfied. It's time to start Step 2.

STEP 2: USE THE SAS MACRO FACILITY IN PROGRAM CONVERSION

There are several ways that the SAS macro facility is used by stored processes. One way is through the use of the predefined macro programs, %STPBEGIN and %STPEND. There are macro parameters that are reserved for use with the %STPBEGIN and %STPEND macro programs. The various ODS options from **DEMO1_TABLE.SAS** fall into this category. You can utilize user-supplied parameter values in your stored process programs. The macro variable that will become the user-supplied parameter is defined in the original program %LET statement, so let's deal with that parameter first.

USER-SUPPLIED PARAMETERS

To keep the new stored process simple, there is only one input parameter, which will be used to provide a REGION value for the WANTREG macro variable. User-supplied parameters, like WANTREG, could have a hard-coded value in the stored process, could be provided using an interface within the client application, or could be provided using a custom-coded interface. In the original program code, the value of the WANTREG macro variable was hard-coded with a %LET statement. In order to change the %LET statement, a programmer or sophisticated end-user must know the valid values for the REGION column. However, in the SAS Intelligence Platform, the stored process author registers the input parameter in the metadata and can provide valid variable values. The end-user is presented with an easy-to-use interface from which to select a valid value.

The advantage of using input parameters is not just the flexibility that allows the stored process author to deliver more generic, re-usable programs. The metadata is the "boss" as far as parameter usage is concerned. It can contain information such as whether the parameter is required and/or modifiable, whether there are parameter value constraints to be enforced by the client interface, and/or whether the user can make only a single selection for parameter values or multiple selections. Figure 4 shows the prompting interface in SAS Enterprise Guide for the WANTREG parameter. The same property sheet interface is used by the SAS® Add-In for Microsoft Office.

Figure 4: SAS Enterprise Guide Interface (with value choices shown)

This parameter interface reflects the way the parameter metadata was defined within SAS® Management Console, as seen in Figure 5.

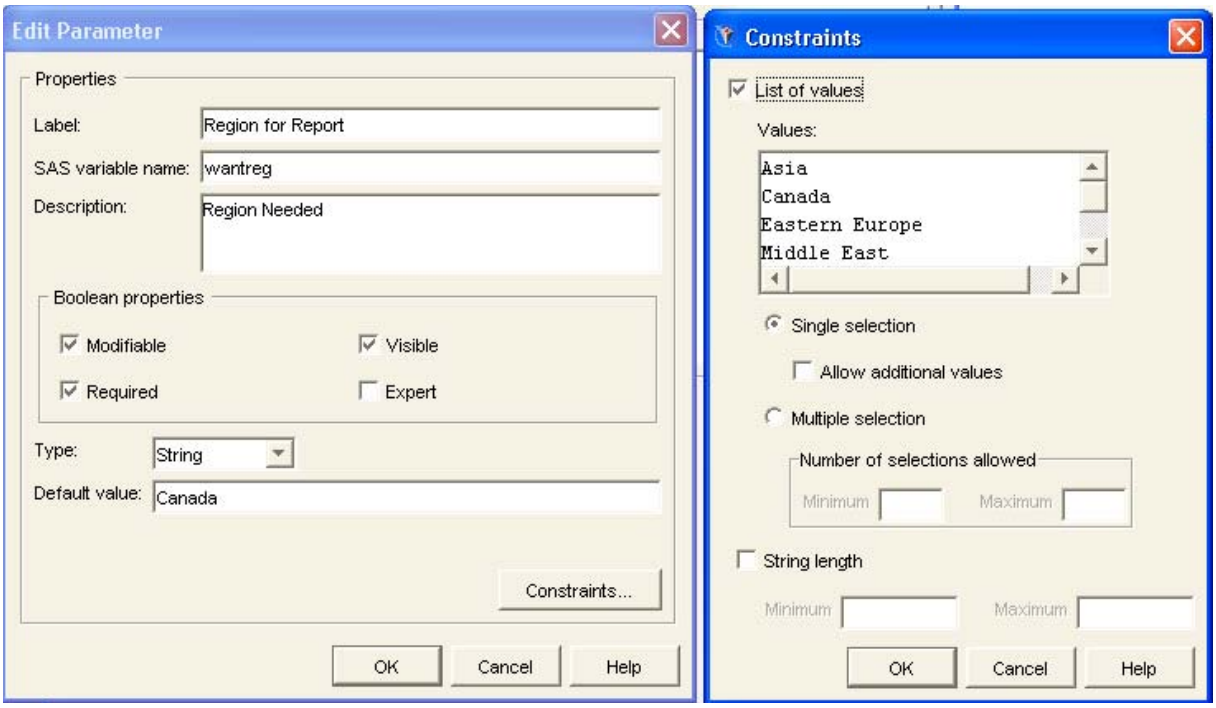

Figure 5: SAS Management Console Parameter Properties and Constraints

In addition to the WANTREG macro variable, more could be done with input parameters, such as providing lists of variables to be analyzed, specifying the type of analysis to be done, and/or specifying presentation options for the report. The possibilities are limited only by your imagination (with a few server-specific limitations).

To finish Step 2, we need to look at the other input parameters—the ODS options that the stored process author has determined are necessary. We also need to talk a bit more about the SAS macro facility. All the input parameters to stored processes are implemented as global macro variables in the stored process program.

NAME/VALUE PAIRS AND INPUT PARAMETERS

What if you're not all that familiar with the SAS macro facility? Here's an analogy to help describe how input parameters work. Input parameters are conceptually like the name/value pairs used in Web applications. For example, if you went to Google and typed "stored processes" in the Search box, you would get a page of hits. Figure 6 shows the first hit at the top of the page.

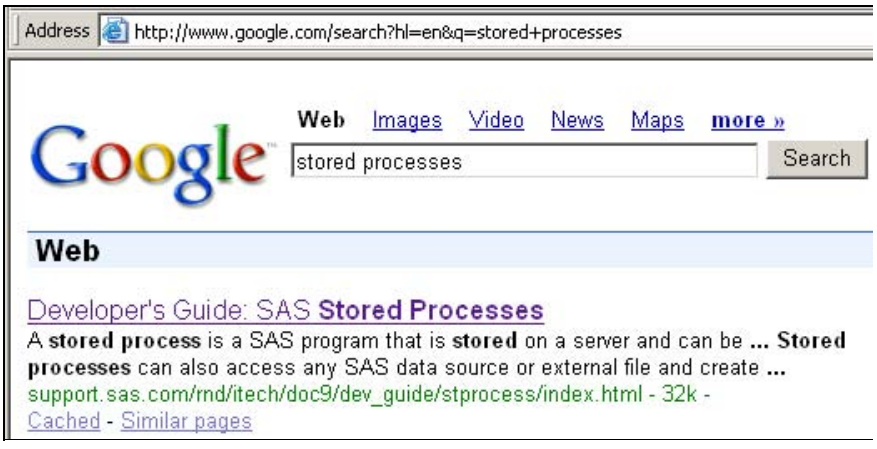

Figure 6: Google Search Results with URL Showing Search Program Parameters

If you look in the browser bar (instead of at the hits), you will see the following URL (Figure 7), which executed when you clicked the Search button:

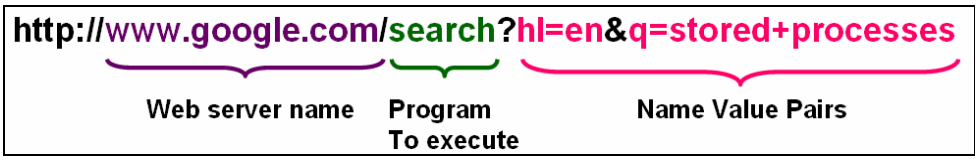

Figure 7: Expanded View of Google Search URL

In the above example, HL and Q are the names of parameters used by the Google SEARCH program. The programmer who wrote the Google SEARCH program had to know that HL and Q were possible parameters that would come from the HTML form interface (the page where you clicked the Search button was a form).

The stored process author has to know what parameters will come from the SAS Intelligence Platform client interface. In a stored process program the name of the parameter is a SAS macro variable name, and the value of the parameter is a SAS macro variable value. The input parameters that you use in your stored process must follow the same rules as for SAS macro variables:

1. Macro variables can be accessed through normal macro syntax (`&macvar`) or through functions like `SYMGET`, `SYMGETC`, or `SYMGETN`. Macro variables can be assigned values using `CALL SYMPUT` or `%LET` or through PROC SQL, in addition to having values collected and set via the client application interface.

2. Parameter names can be no longer than 32 characters and must start with an alphabetic character or underscore and can contain only alphanumeric characters or underscores.

For more information about server-specific rules related to input parameters, refer to the "Input Parameters" section of the *SAS 9.1.3 Integration Technologies Developer's Guide* (SAS Institute Inc. 2007(a)).

In order for the user-supplied input parameters to be used by your stored process, you have to do a little more than just reference them in your program and register them in the metadata. Your stored process has to be able to find the input parameters in the macro global symbol table. This part of using input parameters is discussed next.

%STPBEGIN AND %STPEND MACRO PROGRAMS

The main part of the program conversion takes place when the `%STPBEGIN`; and `%STPEND`; macro invocation statements replace the ODS invocation statements. In addition, although it is not required, the use of the `*ProcessBody;` comment is recommended for maximum flexibility. On the workspace server, macro variables are not initialized until the `*ProcessBody` comment is encountered. The macro variables used by `%STPBEGIN`/`%STPEND` are already in the macro global symbol table. The user-defined input parameters also need to be in the global symbol table. You will want to use a `%GLOBAL` statement to place any user-defined input parameters in the right place.

The `%STPBEGIN` and `%STPEND` macro program calls replace the `ODS HTML` "sandwich"; they are a matched set and, in concept, perform the same functions as `ODS HTML` OPEN and CLOSE statements. The `%STPBEGIN` macro initializes the Output Delivery System to generate output from the stored process. The `%STPEND` macro ends ODS processing and delivers the results to the client application that executed the stored process. These macro programs use many reserved macro variables to control how they operate. Some of these macro variables are available for you to test or even change. Generally, each of the client applications in the SAS Intelligence Platform has a preferred ODS destination or result type. HTML is not the default result type for all of the SAS Intelligence Platform applications, but, conceptually, the `%STPBEGIN;` call sets up the ODS environment for a client application in much the same way that an `ODS HTML` OPEN statement sets up the ODS environment to create an HTML file.

There is an alternate, more advanced, invocation method for stored processes that is generally used with Web-based client applications. This method is very similar to the kind of invocation (using `FILE=_WEBOUT`) that was used with SAS/IntrNet® Application Dispatcher programs. A discussion of this advanced technique is outside the scope of this paper. However, these conversion issues are discussed in the SAS Global Forum 2007 paper "Converting SAS/IntrNet Programs to SAS Stored Processes" (Weinstein 2007).

OVERRIDING RESERVED MACRO VARIABLES USED FOR ODS OPTIONS

When your stored process is executed by one of the client applications, `%STPBEGIN` initializes the macro parameters that are appropriate for the invoking client application. For example, the default result type or ODS destination for SAS Enterprise Guide is HTML format, while the default result type for SAS® Web Report Studio is the SAS Report

(XML) format. The reserved macro variable, _ODSDEST, holds the name of the destination being used. Normally, you do not need to change the _ODSDEST parameter if you want your stored process to generate the result type that is the default for the client application or if you want your end-users to be able to select a result type using client application selection methods. Consider a stored process written based on **DEMO1_TABLE.SAS**. The only changes needed for the program are those shown below. You can download the full text of the modified code in the **DEMO2_HTML.SAS** program from the Web site listed in the "Resources" section at the end of the paper.

Original Code From DEMO1_TABLE.SAS	Modified Code Saved as DEMO2_HTML.SAS

`ods html file='c:\temp\demo1.html'`
`  style=sasweb rs=none;`
 	
`. . . SAS code . . .`
 	
`ods _all_ close;`
	`%global wantreg _odsoptions _odsstyle`
	`_odsstylesheet;`
	`*ProcessBody;`
	`%let _odsoptions = rs=none;`
	`%let _odsstyle = sasweb;`
	`%let _odsstylesheet=;`
	
	`%stpbegin;`
	
	`. . . SAS Code . . .`
	
	`%stpend;`

The %GLOBAL statement ensures that all input parameters are declared, creating an empty macro variable for each possible input parameter. This enables you to reference the macro variable in your stored process program, even if it was not set by the stored process client application. If an input parameter value is not supplied by a client application, declaring it in a %GLOBAL statement prevents unwanted WARNING messages and possible program errors from occurring. This is why the WANTREG macro variable appears in the %GLOBAL statement. The other macro variables, _ODSOPTIONS, _ODSSTYLE, and _ODSSTYLESHEET don't technically need to be in the %GLOBAL statement, since they are already set to global at the client interface side. I prefer to be explicit about their use so that stored process maintenance is easier and subsequent programmers know that I intended to override these reserved macro parameters in my code.

The other macro variables that appear in the %GLOBAL statement are macro variables that are reserved for use with the %STPBEGIN macro. A list of all the reserved macro variables used in the demonstration programs can be found in the Appendix. The most commonly used reserved macro variables are shown in Table 1.

ODS Task or Option	Equivalent Reserved Parameter Name
ODS Destination FILE= (or BODY=)	_ODSDEST This ODS option is not used with %STPBEGIN/%STPEND for most clients. Each client receives the output from the server that executed the stored process. Within each client application there is a way to save the stored process results. There is an alternate invocation method for stored processes in which you could specify FILE=_WEBOUT; however, this invocation method is typically used only for Web-based client applications (an advanced topic).
STYLE=	_ODSSTYLE sets the STYLE= option to an ODS style. The style name that you use must be valid on the server running the stored process. Server sessions may not persist from execution of one stored process to another, so if you create a style template in one stored process with PROC TEMPLATE code, it may not be available to a second stored process running on a different server session unless you write the template to a permanently allocated template store on the appropriate server.
STYLESHEET=(URL=)	_ODSSTYLESHEET sets the Cascading Style Sheet (CSS) to be used in the creation of output (generally, HTML-based output – may not be used by all client applications). In most of our examples, we are turning OFF the use of Cascading Style Sheets (CSS) by issuing a NULL %LET statement.
Other options	_ODSOPTIONS
A complete list of the reserved variables can be found in the "Reserved Macro Variables" section of the *SAS 9.1.3 Integration Technologies Developer's Guide* (SAS Institute Inc. 2007(b)).	

Table 1: ODS Options and Equivalent Parameter Names

If you want to alter any of the reserved macro variables (such as _ODSSTYLE or _ODSOPTIONS) your modifying code needs to appear before the %STPBEGIN; macro program invocation. A stored process can modify the macro variables used by the %STPBEGIN macro program at any time before %STPBEGIN; appears in the stored process code because this is the point at which the %STPBEGIN macro is called. Changing these reserved macro variables after %STPBEGIN has been invoked is not recommended.

The one ODS option from the original program that has not been converted is the FILE= option. The original program created a file called DEMO1.HTML. The SAS Intelligence Platform application dynamically requests and dynamically receives the stored process results in the open SAS Enterprise Guide project, Word document, PowerPoint presentation, Excel workbook, or SAS Web Report Studio report. The person who invokes the stored process has the ability to save the stored process results using the File pull-down menu of the client applications, so the FILE= (or BODY=) option is not needed for this stored process.

Note how the %LET statements for _ODSSTYLE and _ODSOPTIONS precede the %STPBEGIN macro invocation statement in the modified program. _ODSSTYLE is overriding a single ODS option, whose usage is predefined in the %STPBEGIN macro program. There is no pre-defined reserved macro variable for the RS= option; therefore, the %LET statement for _ODSOPTIONS must contain the full specification, RS=NONE.

After the program is converted, it must be registered in the metadata (this is Step 3). Using SAS Management Console, the metadata for the stored process was registered with only the WANTREG parameter and the following general properties and execution environment settings:

Name: demo2_html
Server: SAS Main – Logical Stored Process Server
Source Code Repository: S:\Workshop\winsas\sbisp
Source File: demo2_html.sas
Output Type: Streaming

The names of the servers, such as SASMain, are set by your system administrators. For simple tabular output that returns results to the client application, the Stored Process Server and Streaming are good choices for Server and Output Type, respectively. To reduce confusion, the stored process name is the same as the SAS program name, although in a true production environment, you could just as easily have called this stored process something like "Performance Report".

When the DEMO2_HTML stored process is executed for a SAS Enterprise Guide project and the end-user selected **Canada** for the WANTREG parameter, the partial output would be as shown in Figure 8.

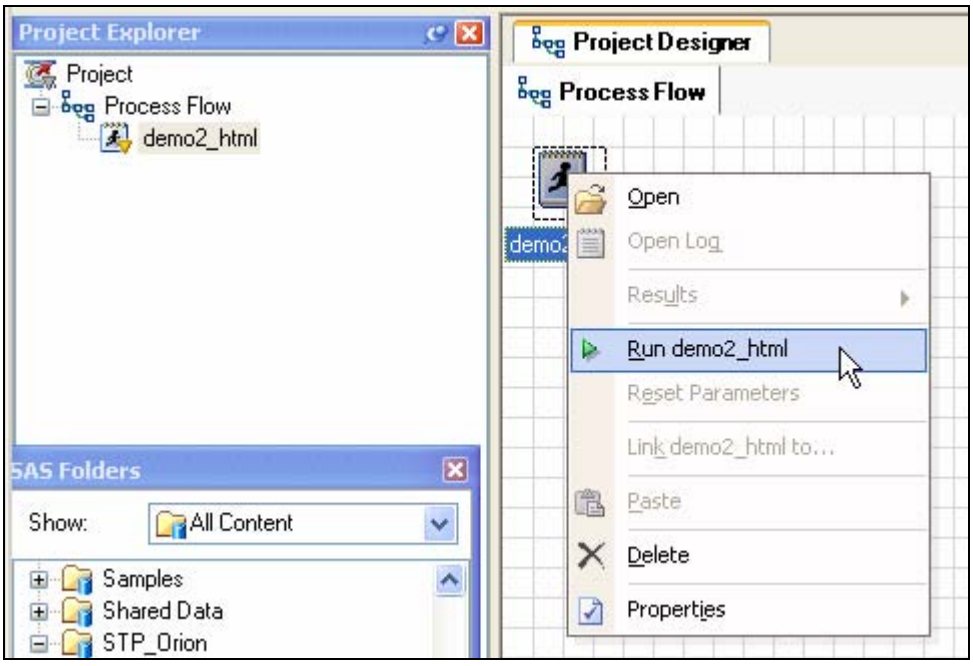

Figure 8: Executing the Stored Process within a SAS Enterprise Guide Project

When the stored process consumer selects **Run** → **demo2_html**, the consumer next sees the parameter property sheet for the WANTREG parameter as shown in Figure 9.

Figure 9: SAS Enterprise Guide Interface (with value choices shown)

If the stored process consumer selects **Canada** (which is highlighted because it is the default value), and then selects **Run**, the output is as shown in Figure 10.

Canada

Region Report

Product	Total Sales
Boot	$385,613
Men's Casual	$441,903
Men's Dress	$920,101
Sandal	$14,798
Slipper	$952,751
Sport Shoe	$140,389
Women's Casual	$410,807
Women's Dress	$989,350
	$4,255,712

Subsidiary Performance Report

	Product								
	Boot	Men's Casual	Men's Dress	Sandal	Slipper	Sport Shoe	Women's Casual	Women's Dress	Total
Subsidiary									
Calgary	$17,720	0	$12,775	$2,886	$5,676	$9,745	0	$12,601	$61,403
Montreal	$40,213	$53,929	$112,009	$3,002	$135,305	$29,435	$24,497	$132,638	$531,028

Figure 10: Partial Output from DEMO2_HTML Stored Process

The program code for the RTF destination is modified in a similar manner, as shown below.

Original ODS RTF Statements From DEMO1_TABLE.SAS	Modified Code Saved as DEMO3_RTF.SAS
ods rtf file='c:\temp\demo1.rtf' bodytitle startpage=no keepn notoc_data; . . . SAS code . . . ods _all_ close;	%global wantreg _odsoptions _odsdest _odsstyle _odsstylesheet; *ProcessBody; %let _odsdest=rtf; %let _odsstyle=rtf; %let _odsstylesheet=; %let _odsoptions = bodytitle startpage=no keepn notoc_data; %stpbegin; . . . SAS Code . . . %stpend;

The stored process that returns RTF output can be submitted from SAS Enterprise Guide or from the SAS Add-In for Microsoft Office, specifically from Microsoft Word. The output when viewed in Microsoft Word is shown in Figure 11.

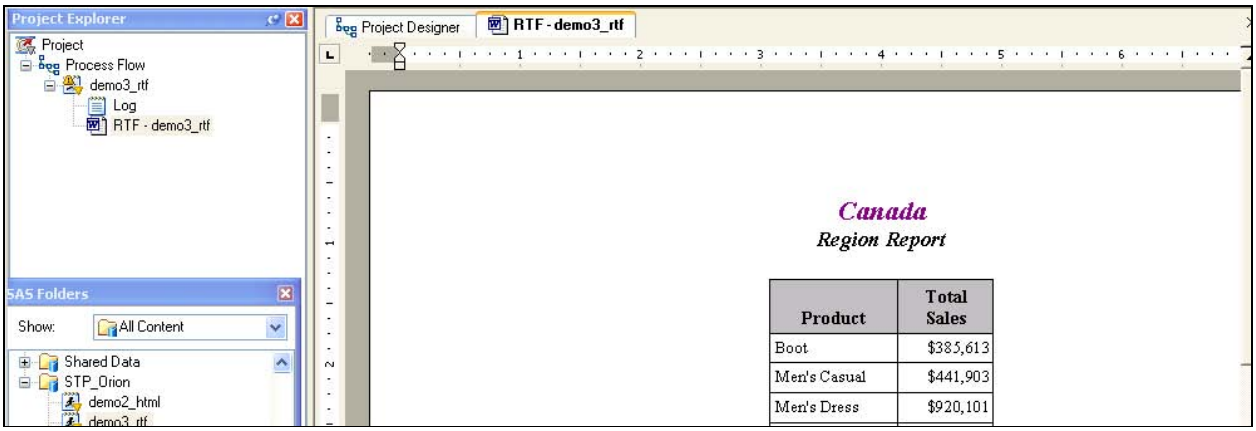

Figure 11: Partial Output from DEMO3_RTF Stored Process

This stored process could also be submitted from the SAS® Information Delivery Portal or from a custom Web application. It could also be executed via URL using the SAS Stored Process Web Application. The stored process that returns RTF would not work, for example, if submitted from Microsoft Excel or SAS Web Report Studio. The result type that you choose for _ODSDEST must be appropriate for the client application from which the stored process will be executed.

The PDF version of the stored process presents some different issues. If the stored process returns PDF results, it can be executed only from SAS Enterprise Guide or via the advanced Web-based methods. It is not appropriate to try to return PDF results to the SAS Add-In for Microsoft Office, or for SAS Web Report Studio. To understand this behavior, try to open a PDF file with Microsoft Word, Microsoft Excel, or Microsoft PowerPoint. If you try to open a PDF file with Microsoft Word, for example, you are prompted about how the file should be converted (Figure 12).

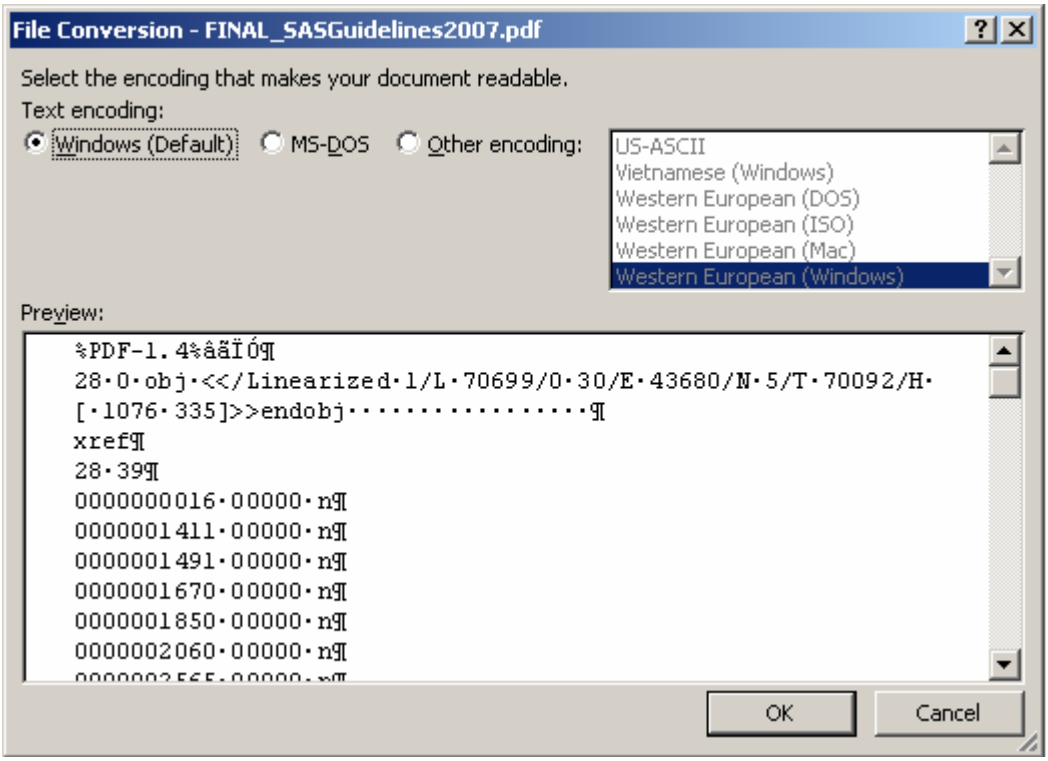

Figure 12: File Conversion Message from Microsoft Word

Essentially, this file conversion message is a clue that PDF files represent a proprietary file type owned by Adobe Systems, Inc. This is the reason that stored processes which return PDF files can be executed only from SAS Enterprise Guide or by using more advanced methods.

To convert the file so it can be executed from within SAS Enterprise Guide we need both the `*ProcessBody;` comment and the `%STPBEGIN` and `%STPEND` macro calls instead of the regular ODS invocation. The program code for the PDF destination is modified in a manner similar to those shown before.

Original ODS PDF Statements From DEMO1_TABLE.SAS	Modified Code Saved as DEMO4_PDF.SAS
ods pdf file='c:\temp\demo1.pdf' bodytitle startpage=no keepn notoc_data; . . . SAS code . . . ods _all_ close;	%global wantreg _odsoptions _odsdest _odsstyle _odsstylesheet; *ProcessBody; %let _odsdest=pdf; %let _odsstyle=printer; %let _odsstylesheet=; %let _odsoptions = bookmarkgen=no compress=9 startpage=no; %stpbegin; . . . SAS Code . . . %stpend;

Now that we have three different stored processes that return the desired results, the only task left is to join them back together in one stored process that could be used for any of the three destinations. Since the purpose of a stored process is primarily to execute one request and return one set of results, it is not possible in a stored process like this one to create multiple output types from one execution of the stored process. In a more advanced scenario

(using permanent result packages), it would be possible to create multiple output files from one execution of a stored process. However, a discussion of this advanced topic is outside the scope of this paper.

Using macro conditional logic, it is, of course, possible to write one stored process program that would execute and return one of the three result types, RTF, PDF, or HTML. While this is an easy macro programming task, it raises several other issues. The issues revolve around how the end-users actually work and what the end-users are allowed to control.

If you are a member of the client-centric school of thought, then, there is no need to control or override the destination or result type, because each of the client applications allows the end-user to set the result type (essentially _ODSDEST) for the stored process results returned to that client application.

On the other hand, the results-centric school of thought holds that the stored process was designed as part of a corporate reporting system where other factors (regulatory requirements, corporate standards) dictate the result type. Therefore, according to this school of thought, it is up to the stored process author to control what values of _ODSDEST are used or to present the end-user with a limited set of choices for _ODSDEST. For example, when you use the BODYTITLE option (which is a requirement for some pharmaceutical-related reports) it is valid only with the RTF destination. If that program did not have an override for _ODSDEST, and the program were submitted from SAS Enterprise Guide, with the default result type set to HTML, the stored process would not run and the SAS log would contain the following error message (highlighted).

```
3   +%global wantreg _odsoptions _odsstyle _odsstylesheet;
4   +
5   +*ProcessBody;
6   +
7   +%let _odsstyle=rtf;
8   +%let _odsstylesheet=;
9   +%let _odsoptions = bodytitle startpage=no keepn notoc_data;
10  +
11  +%stpbegin;
11    bodytitle startpage=no keepn notoc_data
      _________ _________
            22        202
ERROR 22-322: Syntax error, expecting one of the following: ;, (, ANCHOR, ARCHIVE, ATTRIBUTES,
BASE, BODY, CHARSET, CLOSE, CODE, CONTENTS, CSS, ENCODING, FILE, FRAME, GFOOTNOTE, GPATH,
GTITLE,
              HEADTEXT, METATEXT, NEWFILE, NOGFOOTNOTE, NOGTITLE, PAGE, PARAMETERS, PATH,
RECORD_SEPARATOR, STYLE, STYLESHEET, TEXT, TRANTAB.
ERROR 202-322: The option or parameter is not recognized and will be ignored.
```

Both schools can be satisfied by designing a stored process that utilizes conditional macro programming. In the program download, the **DEMO5_MACRO.SAS** program utilizes one macro to set the report options and another macro program to run the report code.

DEMO5_MACRO.SAS
(SAS Macro logic added to DEMO1_TABLE.SAS)

Program Code	Comments
`%global wantreg _odsoptions _odsdest _odsstyle _odsstylesheet;`	Declare GLOBAL macro variables.

`%macro setopt;`
`%if %upcase(&_odsdest) = RTF %then %do;`
`  %let _odsstyle=rtf;`
`  %let _odsstylesheet=;`
`  %let _odsoptions=bodytitle startpage=no nokeepn notoc_data;`
`%end;`
	Define SETOPT macro program.
	RTF settings for SAS Add-In for Microsoft Office.

`%else %if %upcase(&_odsdest) = PDF %then %do;`
`  %let _odsstyle=printer;`
`  %let _odsstylesheet=;`
`  %let _odsoptions=bookmarkgen=no compress=9 startpage=no;`
`%end;`
	PDF settings for SAS Enterprise Guide or Stored Process Web Application.

`%else %if %upcase(&_odsdest) = HTML %then %do;`
`  %let _odsstyle=sasweb;`
`  %let _odsstylesheet=;`
`  %let _odsoptions= rs=none;`
`%end;`
	HTML settings for SAS Enterprise Guide, SAS Add-In for Microsoft Office, Stored Process Web Application.

`%else %if %upcase(&_odsdest)=TAGSETS.SASREPORT11 %then %do;`
`  %let _odsstyle=normal;`
`  %let _odsstylesheet=;`
`  %let _odsoptions=;`
`%end;`
	XML SAS Report format for SAS Web Report Studio and other client applications.

`options nodate nonumber missing='0' orientation=landscape;`	General SAS system options which may not be used by all client applications.
`%mend setopt;`	End SETOPT macro definition.
`%macro RegRept;`	Define REGREPT macro program.

`/*`
`. . . SAS Code from DEMO1.HTML . . .`
`. . . for PROC SORT, PROC FORMAT, . . .`
`. . . PROC REPORT and PROC TABULATE . . .`
`*/`
	SAS Report Code unchanged from DEMO1_TABLE.SAS

`%mend RegRept;`	End REGREPT macro definition.

`*ProcessBody;`
`%setopt;`
`%stpbegin;`
`%RegRept;`
`%stpend;`
	Start the stored process-specific code with *ProcessBody. Set the options before %STPBEGIN, and then invoke the reporting macro program. Finally, end the stored process with %STPEND.

This program can be registered two different ways. If you register the program with just an input parameter for REGION (DEMO5_MACRO stored process), you would use this version of the stored process to allow the end-user to pick the result type using the client application options. On the other hand, if you registered the stored process with input parameters for Region and Destination (DEMO5_MACRO_ALT stored process), you would use this alternative stored process if you wanted to allow the end-use to pick the result type (_ODSDEST value) using the property sheet interface.

STEPS 3-4: REGISTER THE STORED PROCESS METADATA AND TEST

The last topic connects Step 2 to Step 3 and Step4. An understanding of the reserved macro variables used by `%STPBEGIN` and `%STPEND` is necessary for converting your legacy programs to stored processes. In the previous examples, we focused primarily on the input parameters and the reserved macro variables; we did not talk about registering the stored processes, explicitly, although each of the stored process was registered in the metadata to use the Stored Process Server as the server for execution and to use Streaming output as the output type. However, in order to successfully execute the stored process and return results to the client application, you also need an understanding of the different output types that a stored process can return. Both of these requirements converge when we talk about converting a SAS/GRAPH program to a stored process.

CONVERSION OF SAS/GRAPH PROGRAMS

When you convert a SAS/GRAPH program, you have several decisions to make that are unique to the production of graphical output:

- Information needed for Step 2 and program conversion:
  - What device driver should be used to create my output?
  - What graphics options need to be supplied in overrides to the `%STPBEGIN` macro call?

- Information needed for Step 3 and metadata registration:
  - What stored process output type do I choose?

The program **DEMO1_GRAPH.SAS** contains two PROC REPORT steps and one PROC GCHART step that uses `DEVICE=ACTXIMG` to produce the output, as shown below (Figure 13):

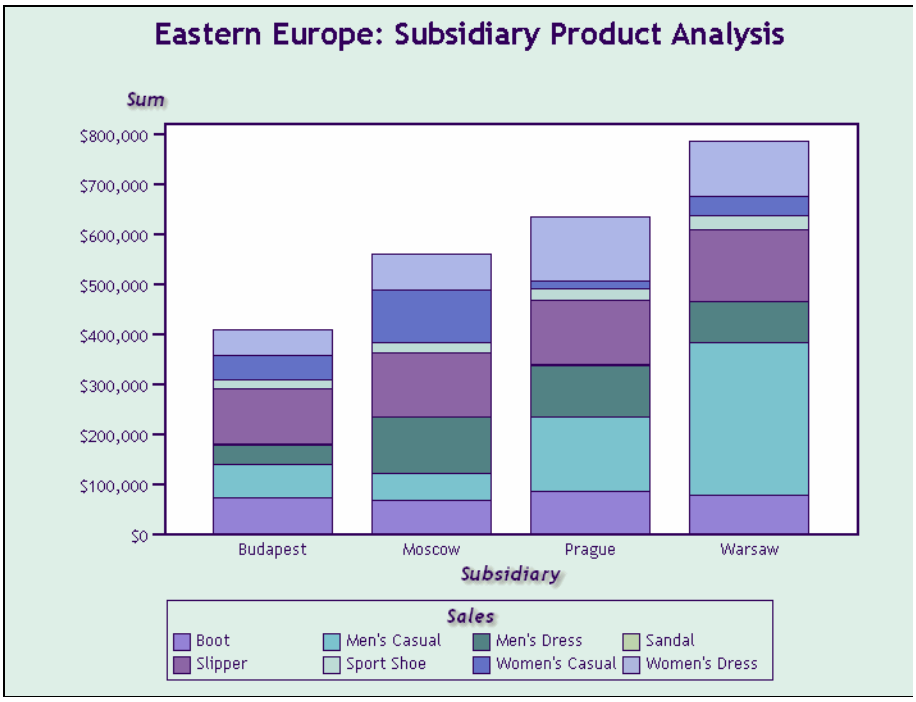

Figure 13: Partial Report Results (SAS/GRAPH only)

Like ODS options, the reserved parameters for graphics options must be specified before `%STPBEGIN`. So the conversion of the program is really no different from the ODS programs above. The conversion for the **DEMO6_GRAPH.SAS** program is shown below.

Original Code From DEMO1_GRAPH.SAS	Modified Code Saved as DEMO6_GRAPH.SAS

ods html path='c:\temp\' (url=none)
file='demo1_graph.html'
style=analysis;
 	
goptions device=actximg xpixels=640
ypixels=480;
 	
. . . SAS Code. . .
ods html close;
	%global wantreg _odsstyle _odsstylesheet
	_odsoptions _gopt_device
	_gopt_xpixels _gopt_ypixels;
	
	*ProcessBody;
	
	%let _odsstylesheet=;
	%let _odsstyle=analysis;
	%let _gopt_device=actximg;
	%let _gopt_xpixels = 640;
	%let _gopt_ypixels = 480;
	
	%stpbegin;
	
	goptions device=&_gopt_device
	xpixels=&_gopt_xpixels
	ypixels=&_gopt_ypixels;
	. . . SAS Code. . .
	%stpend;

Once the program is converted, the stored process is registered in the metadata with the following Execution properties (same as for the previous stored processes) (Figure 14):

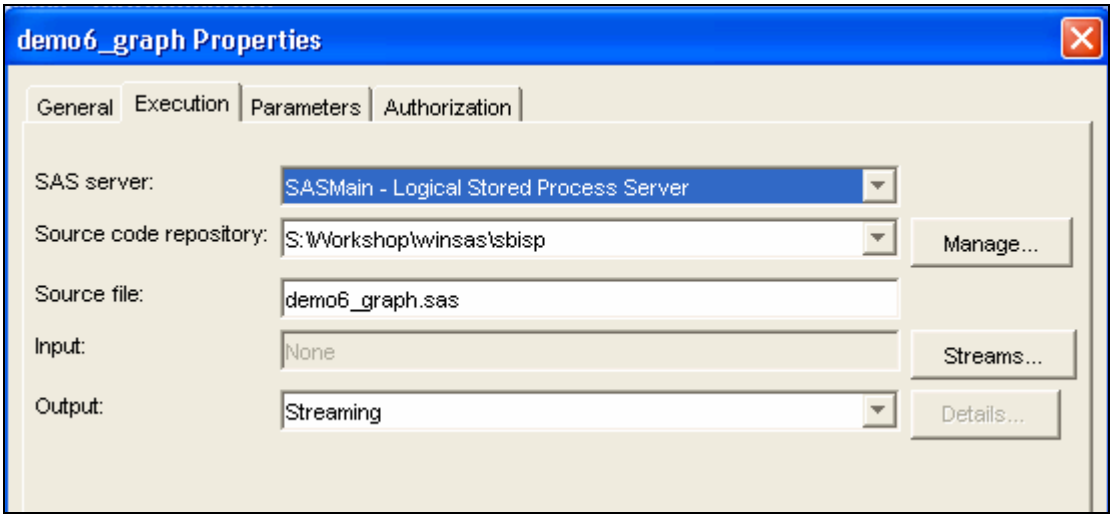

Figure 14: Execution Properties of DEMO6_GRAPH Stored Process

When the stored process executes in SAS Enterprise Guide, the output (Figure 16) looks the same as that shown in Figure 13.

Here's where Step 4 in the conversion process pays off. When we test the stored process from within Microsoft Word, it is immediately apparent from the big X in the Word document (Figure 15) that there was something wrong with the image.

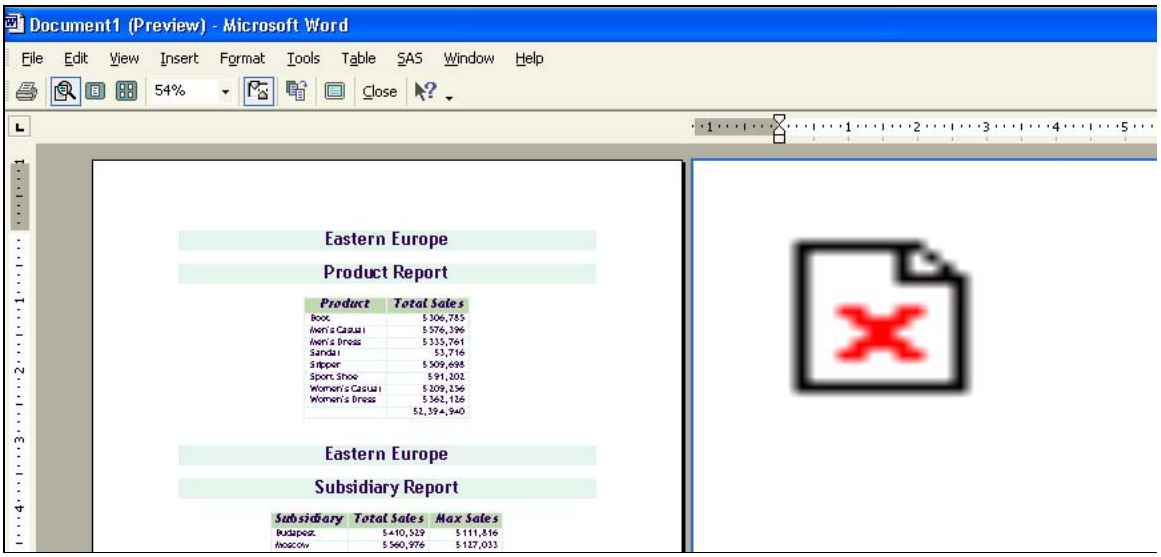

Figure15: Stored Process Results in Microsoft Word When Output Type Is Streaming

Normally, if you have a simple text-based report (such as tabular results), the Streaming output type is fine. However, one of the rules of streaming output is that only one content type can be delivered via the client-server connection. SAS Enterprise Guide takes care of this issue with an option setting that forces Streaming output to be Transient output (Figure 16). Web applications do not have this problem with streaming results because they use the HTTP protocol where graphic output remains on the Web server machine until the client machine has received all the text output. Then another HTTP request goes back to the server for the images, which represent a different content-type.

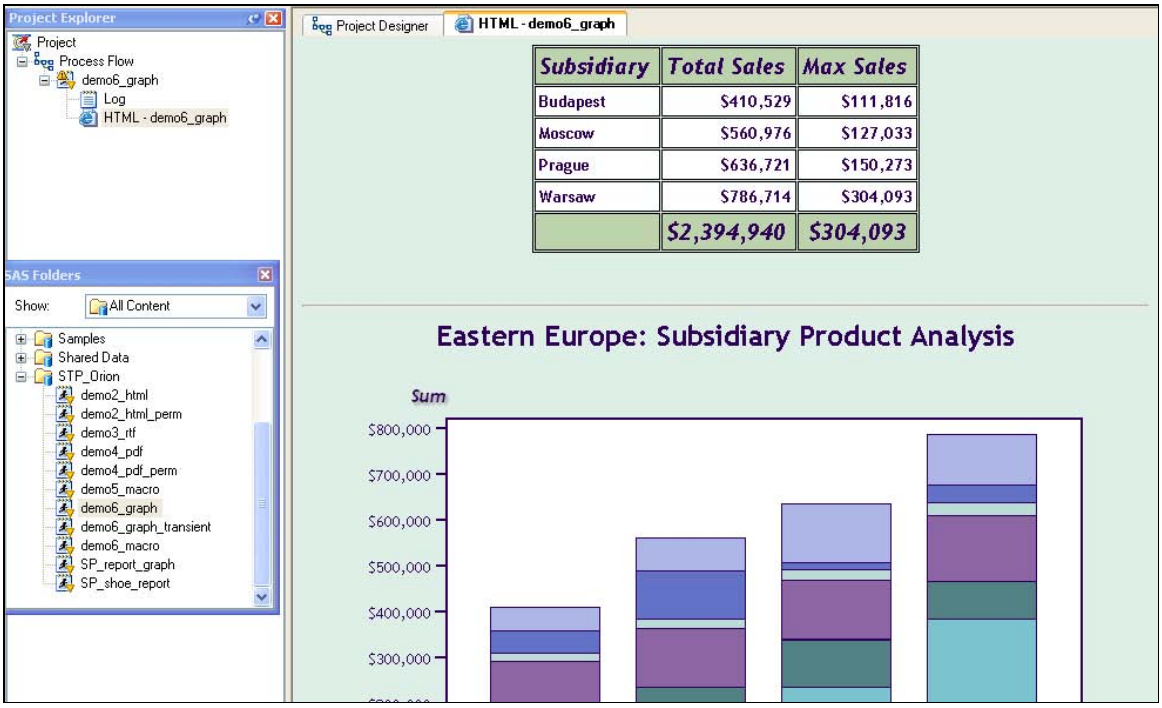

Figure 16: Stored Process Results in SAS Enterprise Guide

Figure 17 shows the reason that the output looked fine in SAS Enterprise Guide and didn't look fine in Microsoft Word. The output type of streaming works well with SAS Enterprise Guide. The "Force Streaming to Transient" setting protects you from the consequences of setting Streaming as the stored process output type.

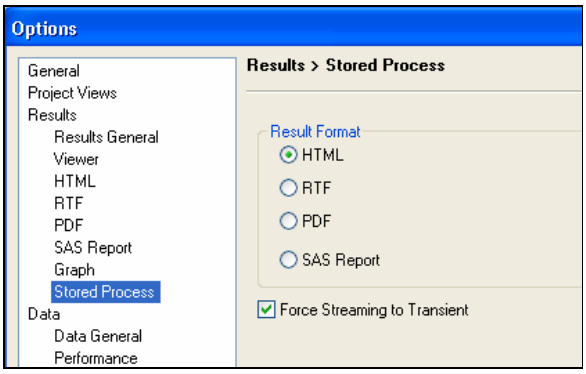

Figure17: Force Streaming to Transient Setting

This setting can be found on the Tools menu. If you select **Tools** → **Options** and look at the default SAS Enterprise Guide settings for a stored process, you will see a setting that forces Streaming output to be generated as Transient output instead. If you turned off this setting (by deselecting it), then SAS Enterprise Guide would also have an X indicating that there were problems with the graphical results.

STREAMING, TRANSIENT, AND PERMANENT PACKAGE RESULTS

The SAS Intelligence Platform can produce two other kinds of output from stored processes instead of Streaming output. Transient output is returned immediately to the client application, and images are stored in a temporary cache location on the client machine. When the client application closes, all the transient output files are cleared. Permanent output is also returned immediately to the client application; however, a stored process that creates Permanent output can either write that output to a permanent location on the server file system or can write the output to a WebDAV repository.

Consider these two different ways to create Transient and Permanent packages using programs that we have already seen (Figures 18 and 19):

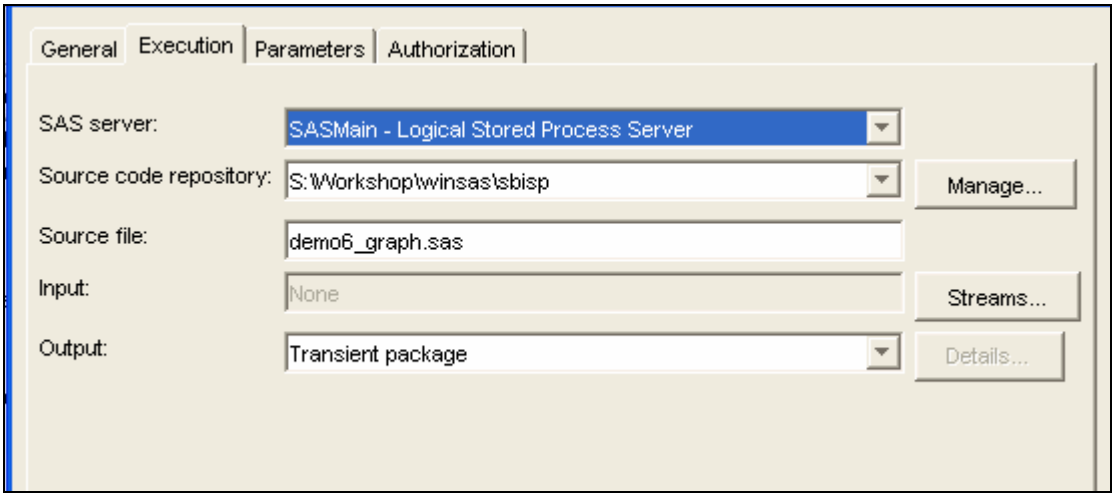

Figure 18: Create Transient Package with DEMO7_GRAPH_TRANSIENT Stored Process

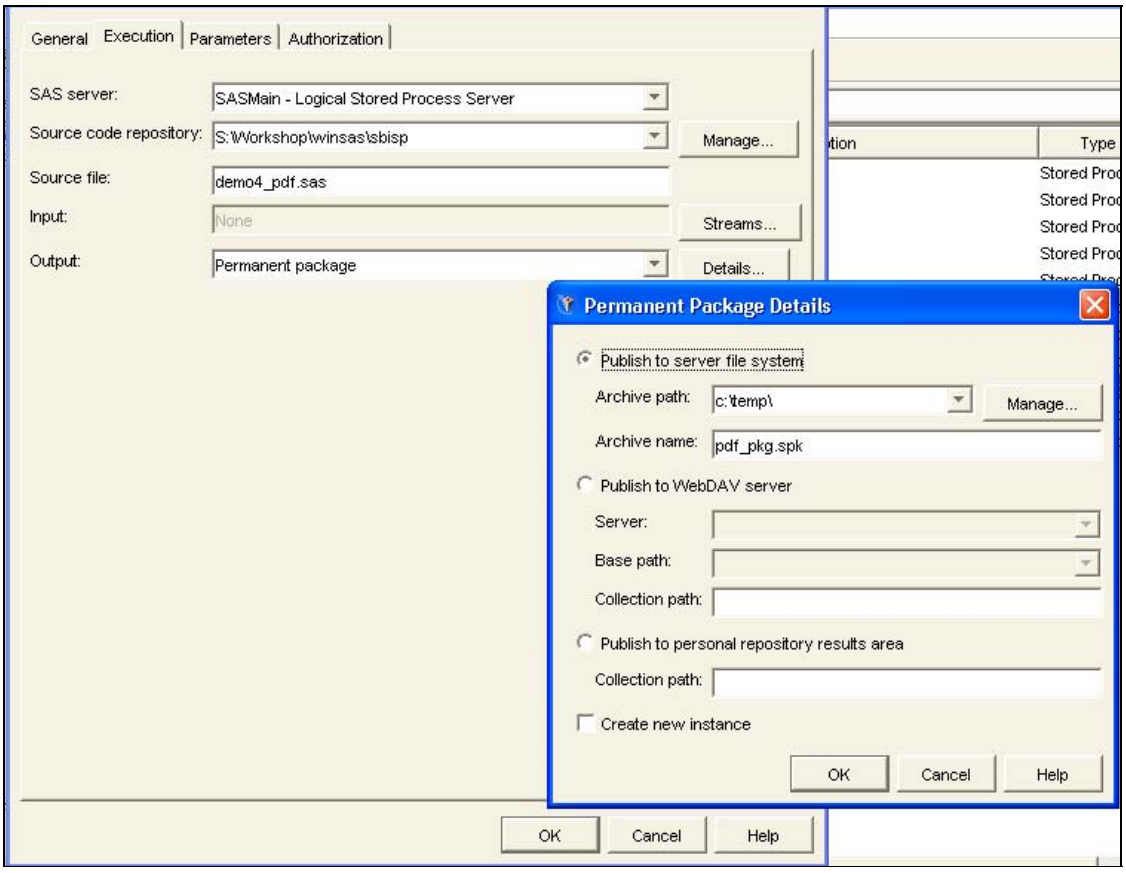

Figure 19: Create Permanent Package with DEMO7_PDF_PERM Stored Process

Note how the converted SAS programs were used as the source files for these stored processes. The **DEMO6_GRAPH.SAS** program was used to produce Figure 20, and the **DEMO4_PDF.SAS** program was used to produce Figure 21. The only change was the use of a different output type; the program code did not change.

The Permanent output is written not as an HTML or PDF file, but instead as a package of files (or in this case, just one file) in a special container called a SAS Package, which is similar to a ZIP file or a TAR file. The SAS Package container permanently stores separate files, which may or may not be the same file type. For an explanation of the possible file types that you can put into a SAS Package, refer to the section "About Packages: Package Content" in the *SAS 9.1.3 Integration Technologies Developer's Guide* (SAS Institute Inc. 2007(c)). For more information about administering and configuring Xythos in order to publish output to a WebDAV server, refer to the section "Implementing Authentication and Authorization for the Xythos WFS WebDAV Server" in the *SAS 9.1.3 Integration Technologies Server Administrator's Guide* (SAS Institute Inc. 2007).

If you run the **DEMO7_GRAPH_TRANSIENT** stored process, then Microsoft Word results would now contain the graphic image as shown in Figure 20:

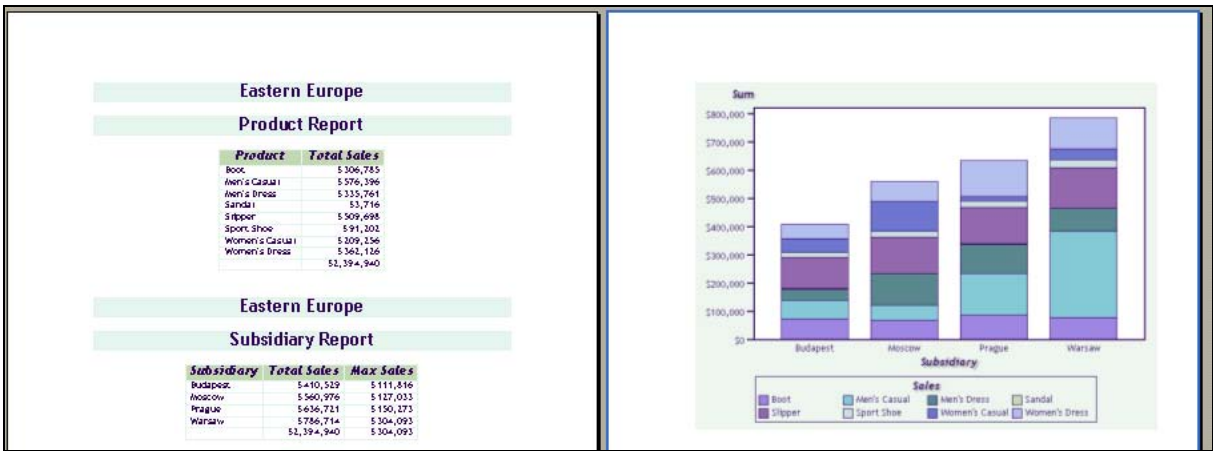

Figure 20: Partial Stored Process Transient Results Shown in Microsoft Word Print Preview

If you run the **DEMO7_PDF_PERM** stored process, then you would see this output in SAS Enterprise Guide (Figure 21):

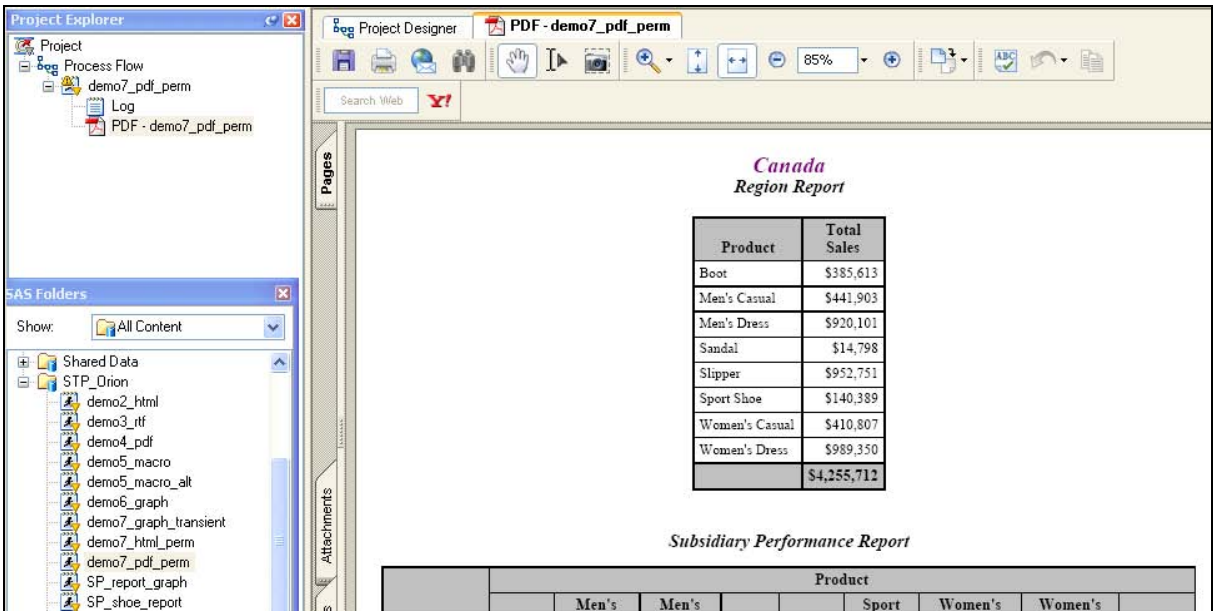

Figure 21: Partial Stored Process Results Shown in SAS Enterprise Guide Project

An examination of the C:\TEMP directory (which is on the server file system for a single machine install of the SAS Intelligence Platform on Windows) reveals an SPK file (Figure 22):

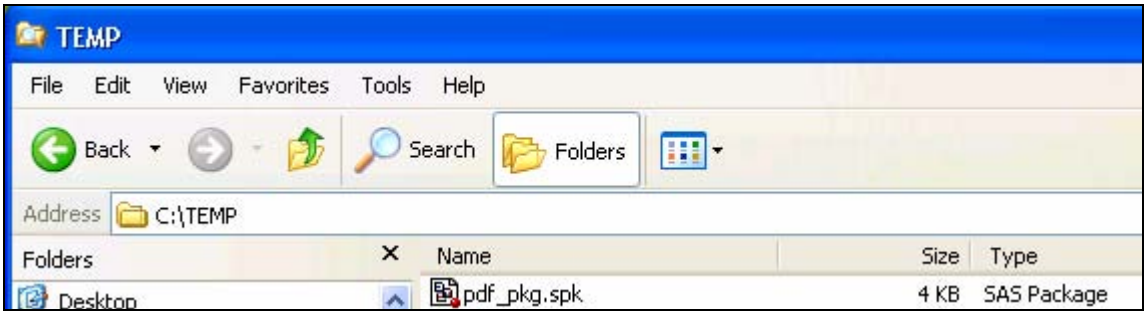

Figure 22: SAS Package on the Server File System

Using the SAS Package Reader to look inside the PDF_PKG.SPK file, you see the PDF file, as shown in Figure 23. If you open the PDF file, it will look the same as the output in Figure 21. The SAS Package Reader will essentially "unpack" the file into a temporary location, which is defined when you install the SAS Package Reader (Figure 23). For more information about the SAS Package Reader (which is freely downloadable and distributable) refer to the SAS Integration Technologies Software download Web site: www.sas.com/apps/demosdownloads/setupcat.jsp?cat=SAS+Integration+Technologies.

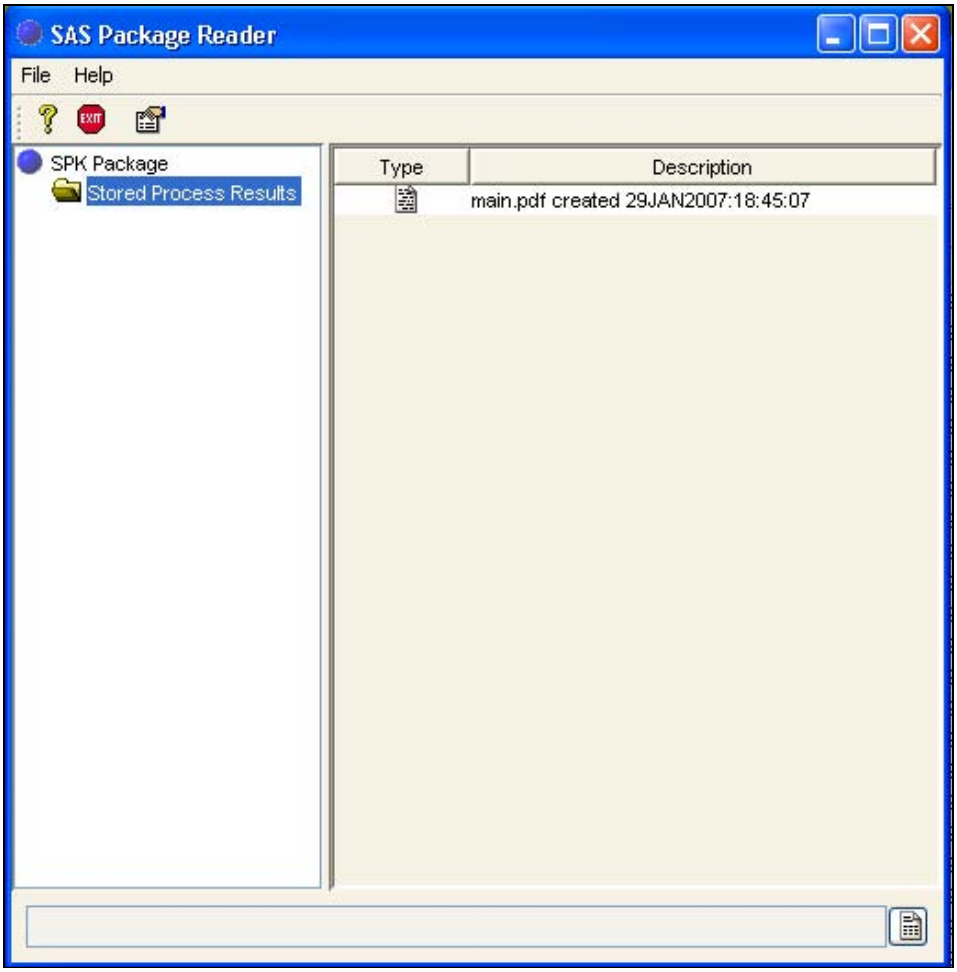

Figure 23: Opening a SAS Package.SPK File with SAS Package Reader

Although this file was written to a directory on the server file system, it could just as easily have been e-mailed to a distribution list or written to a WebDAV or to a personal repository.

CONCLUSION

This paper covered a broad range of topics beyond the basic concepts of using `%STPBEGIN` and `%STPEND` to turn a SAS program into a stored process. We started with a discussion of basic stored process conversion and then covered how to override ODS options and SAS/GRAPH options in the conversion process. The discussion of SAS/GRAPH options and stored processes that returned graphical results led us into the realm of output types and the difference between Streaming, Transient, and Permanent output types.

When you are dealing with legacy programs that use the Output Delivery System, once you understand how reserved macro parameters for `%STPBEGIN` work, it is easy to convert your legacy ODS programs to execute as stored processes. Further study of issues such as publishing to a WebDAV repository or writing a custom Web application (from which you could call a stored process via URL) or using Transient and Permanent packages for stored process output will make your stored processes even more flexible and powerful.

APPENDIX

ODS OPTIONS USED IN THIS PAPER

Option	Destination Used	Description
BODYTITLE	RTF	The BODYTITLE option is an RTF option that instructs ODS to put any SAS TITLE or FOOTNOTE statements into the BODY of the RTF file, as text. Without the BODYTITLE option, ODS uses the RTF control strings for headers and footers around SAS titles and footnotes, so they are repeated on every page of the RTF file when opened in a word processor, such as Microsoft Word.
BOOKMARKGEN=NO	PDF	The BOOKMARKGEN option controls whether PDF bookmarks are generated for a PDF document created by ODS. When you specify NO, no PDF bookmarks are created for the resulting PDF file.
COMPRESS=9	PDF	The COMPRESS option controls the compression of a PDF file to reduce its size. The number you specify indicates the desired level of compression; the larger the number, the greater the compression.
KEEPN	RTF	The KEEPN option controls how tables split at pages. When you specify KEEPN, ODS does not allow a table to split at a page break unless the entire table cannot fit on one page. When you specify NOKEEPN, ODS allows a table to split at a page break. KEEPN minimizes page breaks in tables. However, it might use substantially more paper than NOKEEPN because it issues a page break before starting to print any table that does not fit on the remainder of the page.
NOTOC_DATA	RTF	The NOTOC_DATA option is an RTF option that controls how ODS should treat Table of Contents information, which is generally embedded in the RTF file as hidden text. When you specify NOTOC_DATA, the contents data is not embedded in the resulting RTF file. The default setting is TOC_DATA, which does embed contents data in the RTF file as hidden text.
STARTPAGE=NO	PDF, RTF	The STARTPAGE option controls how page breaks are handled at the beginning of each procedure. When you specify STARTPAGE=NO, ODS suppresses any new pages that would otherwise be inserted into the output. Consult the ODS documentation for a description of the other possible values for the STARTPAGE option.
RS=NONE	HTML	The RS=NONE option produces HTML output that is appropriate for the environment in which you run the SAS job. Frequently, this option is useful when files will be created on one operating system, but stored or displayed on a Web server which runs a different operating system.
STYLE=SASWEB	HTML	The STYLE= option overrides the default style template which ODS would otherwise use to create the HTML output.
TEXT=	PDF	The TEXT= option allows you to insert text strings into your output. This option is very useful when also specifying STARTPAGE=NO. For ODS PDF, the TEXT= option is replacing a TITLE statement that is suppressed by the STARTPAGE option.


Note that many of these options are valid for other destinations than those shown in the table. The above table shows only the options and destinations used for this paper.

PARTIAL LIST OF RESERVED MACRO VARIABLES USED BY %STPBEGIN MACRO PROGRAM

Variable Name	Used By	Description
_GOPT_DEVICE, _GOPT_HSIZE, _GOPT_VSIZE, _GOPT_XPIXELS, _GOPT_YPIXELS	%STPBEGIN/ %STPEND	Sets the corresponding SAS/GRAPH option. See the DEVICE, HSIZE, VSIZE, XPIXELS, and YPIXELS options in "Graphics Options and Device Parameters Dictionary" in the *SAS/GRAPH Reference* in SAS Help and Documentation for more information (SAS Institute Inc. 2006).
_GOPTIONS	%STPBEGIN/ %STPEND	Sets any SAS/GRAPH option documented in "Graphics Options and Device Parameters Dictionary" in the *SAS/GRAPH Reference* in SAS Help and Documentation (SAS Institute Inc 2006). You must specify the option name and its value in the syntax used for the GOPTIONS statement. For example, set _GOPTIONS to ftext=Swiss htext=2 to specify the Swiss text font with a height of 2.
_ODSDEST	%STPBEGIN/ %STPEND	Specifies the ODS destination. The default ODS destination is HTML if _ODSDEST is not specified. Valid values of _ODSDEST need to be appropriate for the client application. Generally speaking, you can return HTML, RTF and PDF results to SAS Enterprise Guide. You can return SASREPORT format XML to SAS Enterprise Guide, the SAS Add-in for Microsoft Office (Excel, Word, PowerPoint). You can also return RTF results to Microsoft Word and HTML results to Word and Excel.
_ODSOPTIONS	%STPBEGIN/ %STPEND	Specifies options to be appended to the ODS statement. Do not use this macro to override options defined by a specific macro variable. For example, do not specify ENCODING=*value* in this variable because it conflicts with _ODSENCODING. **Note:** NOGTITLE and NOGFOOTNOTE are appended to the ODS statement as default options. You can override this behavior by specifying GTITLE or GFOOTNOTE for _ODSOPTIONS.
_ODSSTYLE	%STPBEGIN/ %STPEND	Sets the ODS STYLE= option. You can specify any ODS style that is valid on your system and is available on the server where the stored process executes.
_ODSSTYLESHEET	%STPBEGIN/ %STPEND	Sets the ODS STYLEHEET= option. To store a generated style sheet in a catalog entry and automatically replay it using the SAS Stored Process Web Application, specify myfile.css (url="myfile.css")

For the complete list of Reserved Macro Variables, refer to the section "Reserved Macro Variables" in the *SAS 9.1.3 Integration Technologies Developer's Guide* (SAS Institute Inc. 2007(a)).

FULL TEXT OF STARTER PROGRAMS

DEMO1_TABLE.SAS

```
proc format;
  value incent 0-5000='light yellow'
               .='light yellow'
               50000-high='light green'
               other='white';
run;

%let wantreg=Canada;

ods listing close;
options nodate nonumber missing='0' orientation=landscape;
ods rtf file='c:\temp\demo1.rtf'
        bodytitle startpage=no keepn notoc_data;
ods pdf file='c:\temp\demo1.pdf'
        bookmarkgen=no compress=9 startpage=no;
ods html file='c:\temp\demo1.html'
         style=sasweb rs=none;
ods escapechar='^';

  proc sort data=sashelp.shoes out=shoes;
       by Region;
       where Region = "&wantreg";
  run;

  proc report data=shoes nowd;
    title "^S={Foreground=Purple font_size=16pt}&wantreg";
    title2 "Region Report";
    column region product sales;
    define region /group noprint;
    define product / group;
    define sales /sum f=dollar14.;
    break after region / summarize style=Header;
  run;

  %let ss=^S={just=c font_size=13pt font_weight=Bold font_style=italic};
  ods pdf text="^2n ^_^S={} &ss";
  ods pdf text="&ss.Subsidiary Performance Report";

  proc tabulate data=shoes f=dollar14.;
    title 'Subsidiary Performance Report';
    var sales;
    class subsidiary product;
    table subsidiary all*{s={background=white}},
          Sales*{style={background=incent.}}
               *(product=' ' all='Total'*{S={background=white}})
          / style_precedence=row;
    keylabel sum=' '
             all='Subsidiary Total';
    label Sales='Product';
  run;

ods _all_ close;
```

DEMO1_GRAPH.SAS

```
** these are good regions for the graphs;
%let wantreg=Canada;

ods html path='c:\temp\' (url=none)
         file='demo1_graph.html' style=analysis;
goptions device=actximg xpixels=640 ypixels=480;
ods escapechar='^';
  proc sort data=sashelp.shoes out=shoes;
       by Region;
       where Region = "&wantreg";
  run;

  proc report data=shoes nowd
    style(report)={cellspacing=1};
    title "^S={Foreground=Purple font_size=16pt}&wantreg";
    title2 "Product Report";
    column region product sales;
    define region /group noprint;
    define product / group;
    define sales /sum f=dollar14.;
    break after region / summarize style=Header;
  run;

    proc report data=shoes nowd
         style(report)={cellspacing=1};
    title "^S={Foreground=Purple font_size=16pt}&wantreg";
    title2 "Subsidiary Report";
    column subsidiary sales sales=smax;
    define subsidiary /group ;
    define sales /sum f=dollar14.;
      define smax / max f=dollar14. "Max Sales";
    rbreak after / summarize style=Header;
run;

axis1 minor=none  label=("Subsidiary") ;
axis2 minor=none  label=("Sum");

legend1 frame shape=bar(1,1)cells
        label=(justify=center position=(top center)
        "Sales")
        position=(bottom center);

proc gchart data=shoes;
   title "&wantreg: Subsidiary Product Analysis";
   vbar subsidiary /
        type=sum sumvar=sales
        maxis=axis1 frame
        woutline=1
        raxis=axis2 legend=legend1
        subgroup=product;
run;
quit;
ods html close;
```

REFERENCES

SAS Institute Inc. 2006. "Graphics Options and Device Parameters Dictionary". *SAS/GRAPH Software: Reference*. Cary, NC: SAS Institute Inc.

SAS Institute Inc. (a). 2007. "Input Parameters". *SAS 9.1.3 Integration Technologies Developer's Guide.* Cary, NC: SAS Institute Inc. Available at support.sas.com/rnd/itech/doc9/dev_guide/stprocess/input.html.

SAS Institute Inc. (b). 2007. "Reserved Macro Variables". *SAS 9.1.3 Integration Technologies Developer's Guide.* Cary, NC: SAS Institute Inc. Available at support.sas.com/rnd/itech/doc9/dev_guide/stprocess/reserved.html.

SAS Institute Inc. (c). 2007. "About Packages: Package Content". *SAS 9.1.3 Integration Technologies Developer's Guide.* Cary, NC: SAS Institute Inc. Available at support.sas.com/rnd/itech/doc9/dev_guide/publish/package.html.

SAS Institute Inc. 2007. "Implementing Authentication and Authorization for the Xythos WFS WebDAV Server". *SAS 9.1.3 Integration Technologies Server Administrator's Guide.* Cary, NC: SAS Institute Inc. Available at support.sas.com/rnd/itech/doc9/admin_oma/security/security_accessdav.html.

Weinstein, Heather. 2007. "Converting SAS/IntrNet Programs to SAS Stored Processes". *Proceedings of the SAS Global Forum 2007 Conference*. Cary, NC: SAS Institute Inc.

RESOURCES

You can download the programs and stored process used for this paper from the SAS Web site for Technical Papers and Presentations for SAS Global Forum 2007 at support.sas.com/rnd/papers/.

ACKNOWLEDGMENTS

The author thanks Chris Hemedinger, Stephen McDaniel, and Stacey Syphus for graciously answering all SAS Enterprise Guide questions. In addition, this paper would not have been possible without the encouragement and support of Herb Kirk, Larry Stewart, and Eric Rossland. Thanks are also due to Diane Hatcher, Jane Stroupe, Linda Jolley, Christine Riddiough, and Rick Bell, who helped review this paper.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author:

Cynthia L. Zender
SAS Institute, Inc.
Work Phone: 505-522-3803
E-mail: Cynthia.Zender@sas.com